– Appendix –

A Implementation Details

Additional Data Processing Details. For each data sample, we crop the point cloud obtained from § 4.1 into a local chunk of 102.4m $\times$ 102.4m. The point cloud is then voxelized into the fine-level and coarse-level grids used in § 3.1 with 1024^3 and 256^3 resolutions respectively (with voxel sizes of 0.1m and 0.4m). Our dataset contains 20243 chunks for training and 5380 chunks for evaluation, out of the 798 training and 202 validation sequences.

Input and Evaluation Details. Waymo dataset provides 5 views for each camera frame, namely `front`, `front-left`, `front-right`, `side-left` and `side-right`. However, not all of the baseline methods we compared with in § 4.2 can handle the unconventional camera intrinsic in the `side-left` and `side-right` views. We hence only use the first three views (with a resolution of 1920×1280) in § 4.2 for both the input and the evaluation metrics. However, in § 4.3 we opt to use all 5 views for the input to both our method and the baseline due to compatibility and maximized performance.

For the baselines, the original PixelSplat [4] method does not have depth supervision. To make the comparison fair, we attempt to add a depth supervision loss to it. However, the experimental result shows that the additional loss hurts the performance as shown in Tab. 4. We thus report the results of vanilla PixelSplat in the main paper.

	$T+5$			$T+10$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelSplat [4]	**20.11**	**0.70**	**0.60**	18.77	**0.66**	**0.62**
PixelSplat [4] w/ Depth Supervision	19.91	0.58	0.66	**18.87**	0.56	0.67

Table 4: **Comparison of PixelSplat and PixelSplat with Depth Supervision.**

Training Details. The diffusion loss in Eq (2) is defined similar to [16, 39] with a $\boldsymbol{v}$-parametrization as:

$$\mathcal{L}_{\text{Diffusion}} = \mathbb{E}_{t,\mathbf{X},\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[\left\|\boldsymbol{v}(\sqrt{\bar{\alpha}_t}\mathbf{X} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) - (\sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{X})\right\|_2^2\right], \tag{7}$$

where $\boldsymbol{v}(\cdot)$ is the diffusion network, t is the randomly sampled diffusion timestamp, and $\bar{\alpha}_t$ is the scheduling factor for the diffusion process, whose details are referred to in [16].

We train all of our models using the Adam [23] optimizer with $\beta_1 = 0.9$ and $\beta_1 = 0.999$. We use PyTorch Lightning [10] for building our distributed training framework. For the voxel grid reconstruction stage, we train both coarse-level and fine-level voxel latent diffusion models with $64\times$ NVIDIA Tesla A100s for 2 days. For the appearance reconstruction model, we train it using $8\times$ NVIDIA Tesla A100s for 2 days. Empirically, we use $\lambda = 1.0$ for $\mathcal{L}_{\text{Depth}}$ in Eq (2). Additionally, we use $\lambda_1 = 0.9$, $\lambda_2 = 1.0$, $\lambda_{\text{SSIM}} = 0.1$ and $\lambda_{\text{LPIPS}} = 0.6$ in Eq (6). For image condition, we set the feature channel $C = 32$, the number of depth bins $D = 64$, $z_{\text{near}} = 0.1$ and $z_{\text{far}} = 90.0$. We linearly increase the interval of depth bins.

B Network Architecture

Voxel Grid Reconstruction. We follow [39] to implement the Sparse Structure VAE and the Diffusion UNet. Hyperparameters for training them are listed in Tab. 5 and Tab. 6. We pass the images to distilled DINO-v2 [33] ViT-B/14. We use four 2D convolutional layers (channel dims: $[768, 256, 256, 32, 32]$, kernel size: 3, stride: 1) to further process the DINO-v2 output to predict the image feature and the depth distribution.

Appearance Reconstruction. We process the original input images with three 2D convolutional layers (channel dims: $[3, 16, 32, 32]$, kernel size: 3, stride: $[1, 1, 2]$). For the last two convolutional

	Waymo $64^3 \rightarrow 256^3$	Waymo $256^3 \rightarrow 1024^3$
Model Size	14.9M	3.8M
Base Channels	64	32
Channels Multiple	1,2,4	1,2,4
Latent Dim	8	8
Batch Size	32	32
Epochs	50	50
Learning Rate	1e-4	

Table 5: **Hyperparameters for VAE.**

	Waymo - 64^3	Waymo - 256^3
Diffusion Steps	1000	
Noise Schedule	linear	
Model Size	728M	83.0M
Base Channels	192	64
Depth	2	
Channels Multiple	1,2,4,4	1,2,2,4
Heads	8	
Attention Resolution	16	32
Dropout	0.0	0.0
Batch Size	512	256
Iterations	40K	20K
Learning Rate	5e-5	

Table 6: **Hyperparameters for voxel latent diffusion models.**

layers, we set the residual connections. We additionally positionally encode each voxel and then concatenate the positional encoding [31] of each voxel with the corresponding voxel feature after ray casting. We then apply a 3D sparse UNet to output per-Gaussian parameters. We use GT voxels in appearance reconstruction training. Hyperparameters of this 3D sparse UNet are listed Tab. 7.

Model Size	Base Channels	Channels Multiple	Batch Size	Epochs	Learning Rate
4.3M	32	[1, 2, 4]	32	15	1e-4

Table 7: **Hyperparameters for 3D sparse UNet in appearance reconstruction stage.**

Sky Panorama for Background. For the sky panorama model, we set $H_p = 768, W_p = 1536$ in the training stage and increase $H_p = 1024, W_p = 2048$ in the inference time. To decode sampled sky features into the RGB image, we utilize a 2D CNN network reducing the channel from 32 to 16 to 3 with stride 1, keeping the spatial resolution unchanged.

C SCube+ without Per-scene Training

In § 3.3 we introduce a GAN postprocessing module to refine the rendered images, which is finetuned on each scene. To further improve the efficiency of our method, we hereby present a postprocessing module that is jointly trained on the full dataset, without the need of per-scene finetuning. Specifically, we replace the original GAN with a pix2pix-turbo model [34] (which we denote as SCube+*) and train it with image pairs inferred from our model and the ground truths. The results are shown in Fig. 9. This improved model not only reduces the voxel block artifacts, but also resolve the ISP inconsistencies within the image. After enabling this module, the FPS drops from 138 to 20 but can still be visualized interactively.

	Reconstruction (T)			Prediction ($T+5$)			Prediction ($T+10$)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SCube	<u>25.90</u>	<u>0.77</u>	0.45	19.90	<u>0.72</u>	0.47	18.78	<u>0.70</u>	0.49
SCube+	**28.01**	**0.81**	**0.25**	**22.32**	**0.74**	**0.34**	**21.09**	**0.72**	**0.38**
SCube+*	22.59	0.68	<u>0.38</u>	<u>20.37</u>	0.66	<u>0.41</u>	<u>19.65</u>	0.65	<u>0.42</u>

Table 8: **Quantitative Comparisons on 3D Reconstruction.** The metrics are computed both at the input frame T and future frames. ↑: higher is better, ↓: lower is better.

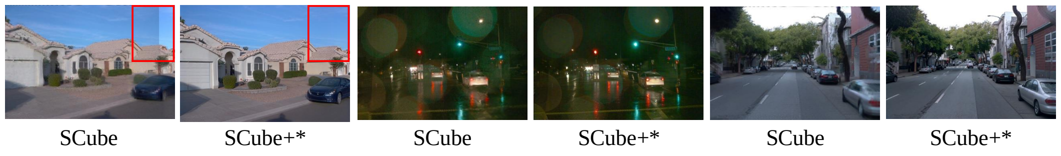


Figure 9: **SCube+*.** Results from the postprocessing network without per-scene optimization. White-balance inconsistencies from different views (marked in red box) can be fixed.

D Additional Results

In this section, we provide more qualitative results on all datasets. We additionally provide a **supplementary video** in the accompanying files to better illustrate our results.

D.1 Geometry Quality

We note that the uncertainty of the scene geometry given our input images is large, and the problem that the model tackles is indeed non-trivial and sometimes even ill-posed. To demonstrate this, we compute the percentage of occluded voxels (that are invisible from the input images) w.r.t. all the ground-truth voxels, and the number is around 80%. To quantitatively evaluate the geometry quality, we compute an additional metric called 'voxel Chamfer distance' that measures the L2-Chamfer distance between the predicted voxels and ground-truth voxels (that are pixel-aligned), divided by the voxel size. This metric reflects the geometric accuracy of our prediction by measuring on average how many voxels is the prediction apart from the ground truth. The results on Waymo Open Dataset are shown in Tab. 9.

Quantile	0.5 *(median)*	0.6	0.7	0.8	0.9
Ours	0.26	0.28	0.32	0.37	0.51

Table 9: **Geometry Quality Comparison.** We show the voxel Chamfer distance comparison between our two-stage model and a single-stage non-diffusion model.

Tab. 9 indicates that on 90% of the test samples, the predicted voxel grid is only half of a voxel off from the ground truth. We note that during our data curation process, there could be errors in the ground-truth voxels (*e.g.*, due to COLMAP failures), accounting for the outliers in the above metric. In the meantime, we visualize the sample with the worst voxel Chamfer distance in Fig. 10. The predicted results are decent even though the ground truth is corrupted due to the lack of motion in the ego car. This demonstrates the robustness of our method.

D.2 Visual Ablation Study

In addition to the quantitative ablation study in Tab. 3, we present a qualitative demonstration in Fig. 11. For the single-stage model, we test the upper bound of it by feeding the ground-truth 1024^3 voxel grids because otherwise the fully-dense high-resolution condition will lead to out-of-memory. The qualitative results match the numbers, showing the importance of using higher-resolution voxel grids and the two-stage model.

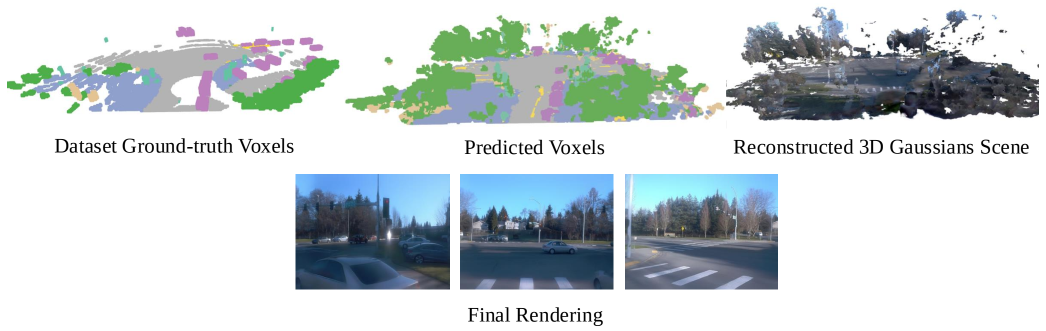


Figure 10: Result on the data sample with the worst voxel Chamfer distance. We show geometry reconstruction and the image renderings.

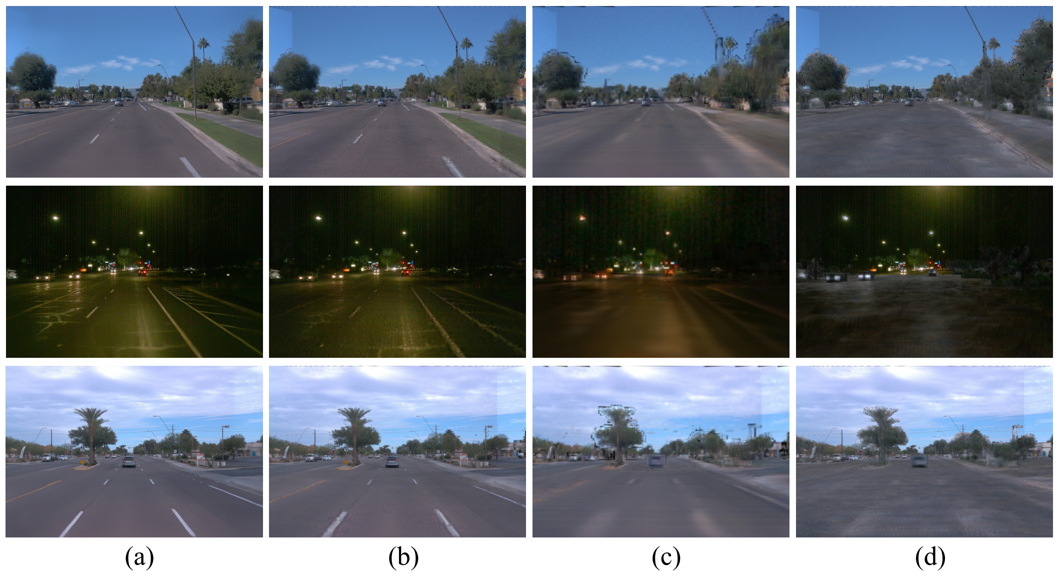
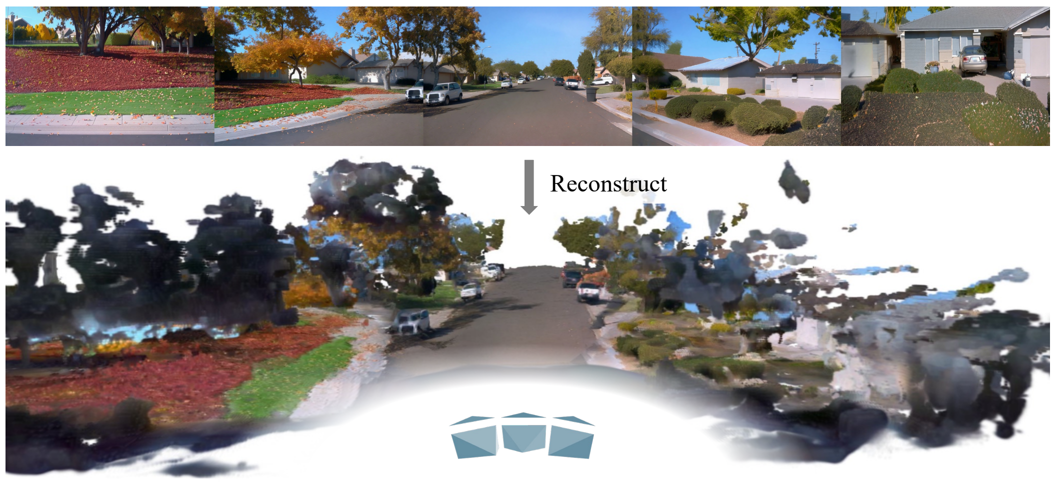


Figure 11: **Visual Ablation Study.** (a) SCube+ (b) SCube (c) SCube with a 256^3 resolution input grid (d) Single-stage model. Zoom in for a better view.

D.3 Additional Results on Text-2-Scene Generation

We provide additional text-2-scene generation results in Fig. 12 and Fig. 13.

A residential neighborhood features houses with well-maintained gardens, autumn-colored trees, lawns with scattered leaves, parked cars, driveways, and clear blue skies.



A residential area features multiple houses, some with specific decorations and vehicles parked outside, including a white pickup truck and a gray car, along with various greenery and utility elements.

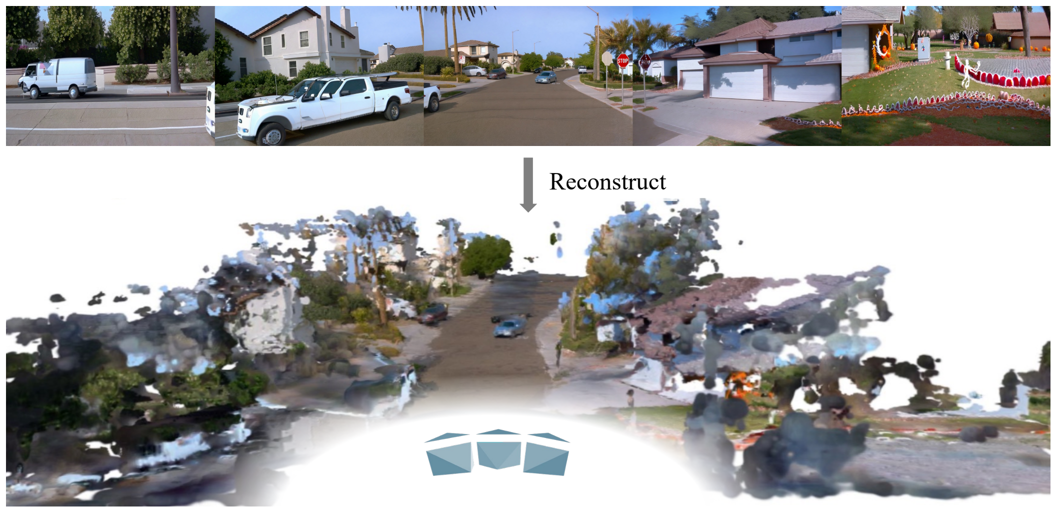
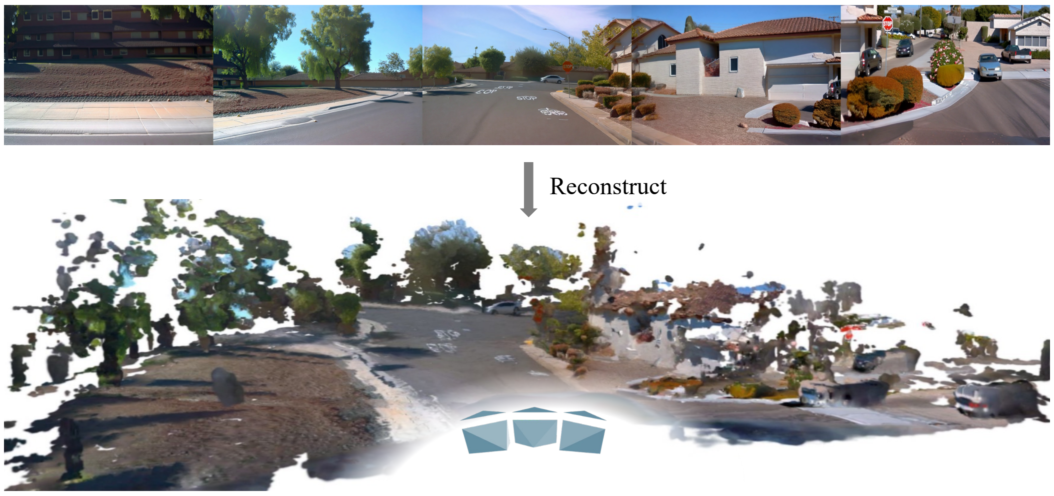


Figure 12: **More Text-2-Scene Generation.** The generated multi-view images may contain flaws, while SCube is still able to reconstruct the 3D scenes.

A suburban neighborhood features two-story houses with reddish-brown roofs and beige walls, marked roads, various parked vehicles, stop signs, and a mixture of gravel, rocks, and trees providing shade on a sunny day.



A suburban neighborhood features a park with green trees, residential houses with red-tiled roofs, streets with bike lane signs and white markings, well-maintained lawns, and sidewalks.

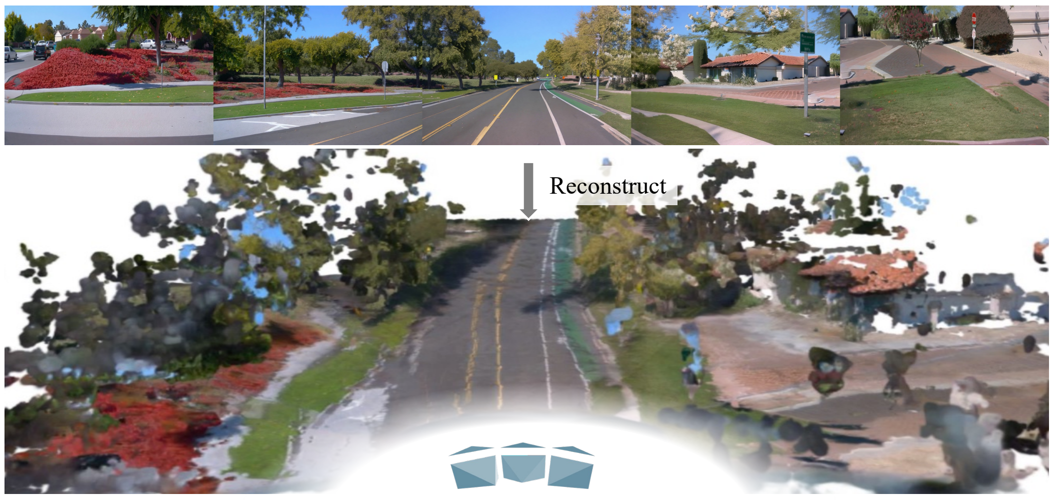


Figure 13: **More Text-2-Scene Generation.**

D.4 Additional Results on Large-scale Scene Reconstruction

We provide additional results on large-scale scene reconstruction from real-world captures in Fig. 14.

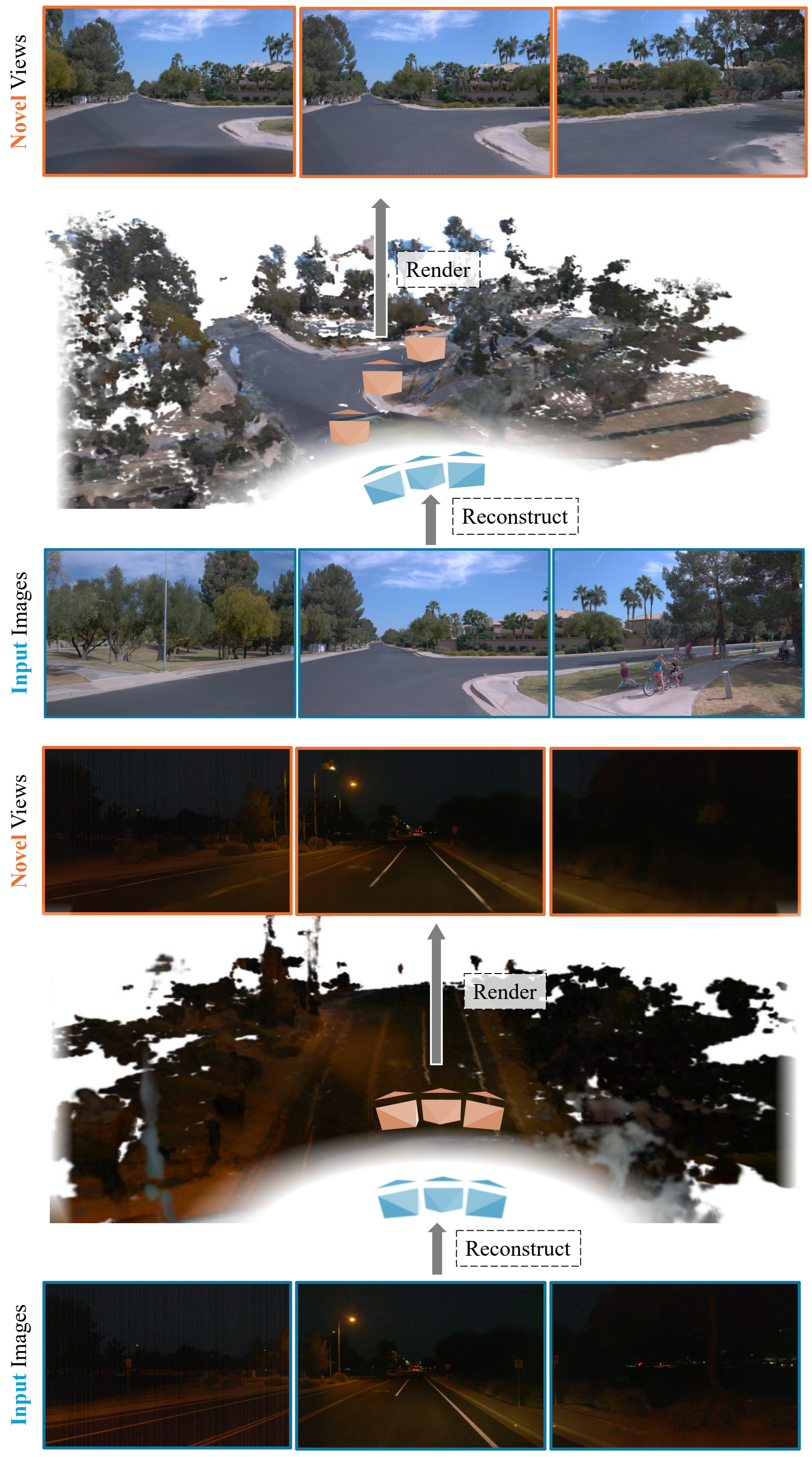


Figure 14: **More Novel View Synthesis.** Our method is able to synthesis extreme novel views.

D.5 Additional Results on LiDAR Simulation

We provide additional LiDAR Simulation results in Fig. 15. We also show the result on a long sequence input in Fig. 16.

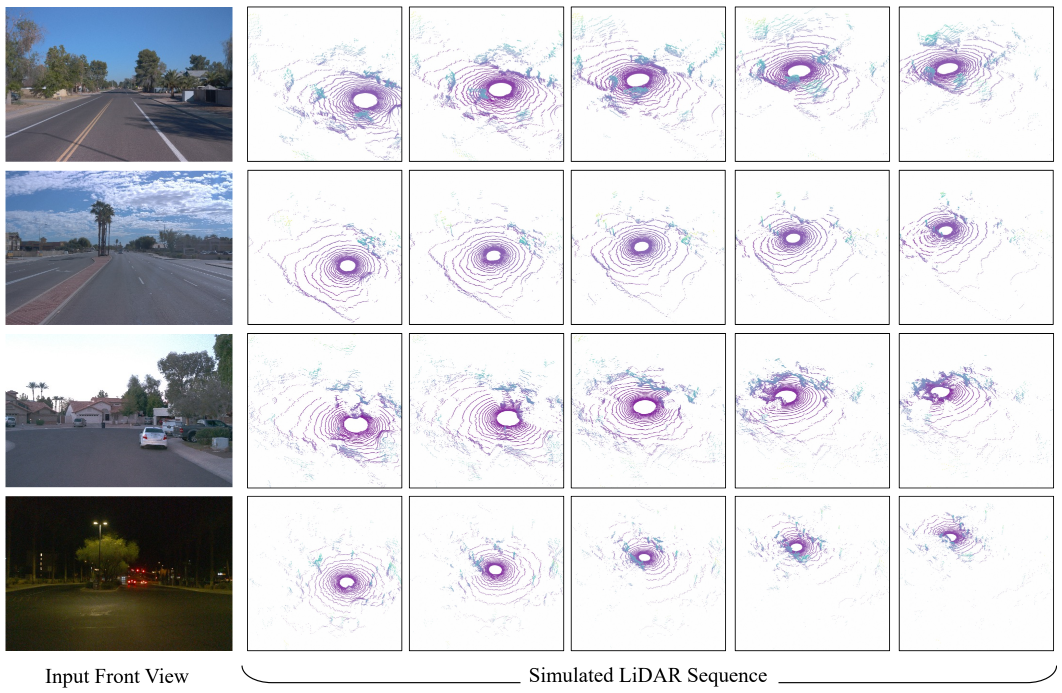


Figure 15: **More LiDAR Simulation results.**

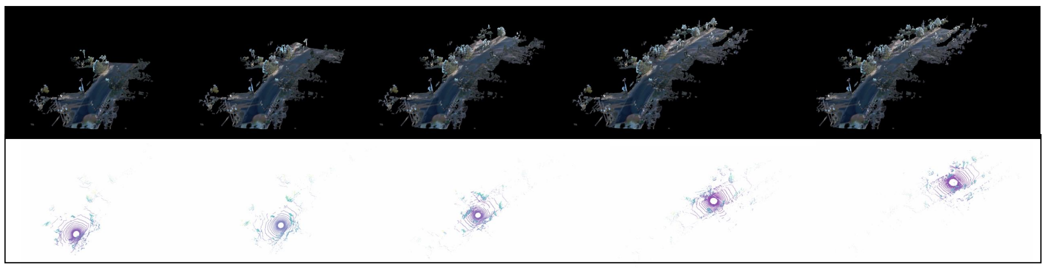


Figure 16: **SCube with Long Sequence Input.** Up: reconstructed scene with appearance. Down: LiDAR simulation result. We chunk the long sequence into clips and apply out method iteratively.